Machinery Control System using Autonomous Agents

Hidehiko Wada
Yokogawa Electric Corporation
2-9-32 Naka-cho, Musashino-shi
Tokyo, 180-8750 JAPAN

Yuichi Sakuraba
Yokogawa Electric Corporation
2-9-32 Naka-cho, Musashino-shi
Tokyo, 180-8750, JAPAN

Masako Negishi
Yokogawa Electric Corporation
2-9-32 Naka-cho, Musashino-shi
Tokyo, 180-8750, JAPAN

Hidehiko_Wada@yokogawa.co.jp Yuiichi_Sakuraba@yokogawa.co.jp Masako_Negishi@yokogawa.co.jp

ABSTRACT

Flexibility will be one of the key features in future manufacturing systems because the environments that surround manufacturing systems are rapidly changing from day to day. To satisfy this requirement, the hierarchical systems used in previous manufacturing systems are not enough because they cannot deal effectively with unexpected situations. Moreover, a lot of software modules have to be modified when a system is rebuilt to meet new requirements. To achieve flexibility in manufacturing systems, the concept of an autonomous decentralized system is useful. We introduce some agents that work autonomously in the system to build autonomous decentralized manufacturing systems.

In manufacturing systems using autonomous agents, the basic concept should be based on a target product and on work having some information moving around in the system. To realize this concept, we introduce mobile agents called product agents as data carriers. The product agents have target product-related information involving manufacturing procedures and data. Each product agent selects a target machine to process and moves on processing machines or controllers according to the specific procedures. The product agents act as autonomous entities in the system.

To verify our proposed concept and architecture, we developed a prototype system. This system was for the Shape Deposition Manufacturing Laboratory of the Robotics Institute at Carnegie Mellon University.

In this paper, we describe our basic concept and the software architecture of our proposed system and explain the prototype system. Then, we include some discussion of our proposed and developed system.

Keywords

Mobile Agent, Autonomous Decentralized Systems, Manufacturing Control Systems.

1. INTRODUCTION

The environments that surround manufacturing systems are rapidly changing because of various consumer needs, sudden changes in market structure, and exchange rates that fluctuate daily [1]. Manufacturing systems have some requirements in order to cope with such drastic changes. These requirements are allowing factories to expand globally, and are providing optimal quality, cost, and on-time delivery of various products. In other words, a manufacturing system that provides variable-type and variable-volume production is needed. The flexibility to cope with such new demands is strongly needed.

Flexibility here means, for example, that production lines can be easily changed, or that the system can cope with individual requests even for the improvement of products, or for custom-made products. Current manufacturing systems cannot satisfy these requirements. To satisfy the requirements, the hierarchical systems used in previous manufacturing systems have not been enough because the systems cannot deal effectively with unexpected situations. Moreover, a lot of software modules must be modified when the system is rebuilt to meet the latest requirements. In other words, current hierarchical manufacturing systems have a rather "stiff" structure.

To achieve flexibility in manufacturing systems, the concept of an autonomous decentralized system is one prospective approach. We introduce some agents that work autonomously in the system to build an autonomous decentralized type of manufacturing system. To satisfy the requirement of constructing flexible systems, we have established the basic concept that information on a target product or work (for example, a procedure to provide product or process data on a product) is attached to the target work and moves with the work. The work is done through direction from the work side instead of central control mechanism side, therefore some central control mechanisms can be removed from the system.

To demonstrate or verify the validity of our idea, we developed an experimental machinery control system[2] for the Shape Deposition Manufacturing (SDM) Laboratory[3] of the Robotics Institute at Carnegie Mellon University (CMU).

We will explain our basic concept and system design in Section 2. In Section 3, we will present the design and implementation of the prototype system at CMU. Section 4 discusses flexibility of our proposed system and the prototype system. Then, we will compare our work with some related work and conclude this paper with future work based on lessons from the prototype system.

2. System Design

2.1 Basic Concept

To realize flexible manufacturing systems, we propose a flat architecture manufacturing system using agents rather than hierarchical systems. The following is our basic concept for the system. An agent is generated for each planned product, and we call these agents "product agents". At the same time, we create an agent for every machine tool or processing device, and we call these agents "machine agents". The machine agents are abstraction for each tool or device. We give the product agents the procedures to make the product in the charge of the product agent.

According to any given order of production steps, the product agent selects an appropriate machine tool and directs the machine tool or controller on how to process the product and lets the machine tool start processing. Actually, the product agents direct the machine agent in charge of the machine tool, and the machine agent controls the controllers or devices of the tool.

The product agent acts autonomously by understanding system status and deals effectively and dynamically with the situations, and completes the product in charge. The agents may negotiate with other agents if necessary.

To realize the concept described above, we design the product agents to be mobile agents and the machine agents to be stationary agents. Machine agents run on controllers which control machine tools or on computers connected to the controllers. On the other hand, product agents are mobile agents and move to the computer where the target machine agent is running following the movement of the target product or work. After moving to the target computer, the product agent sends a process request to the machine agent, and waits until the machine agent makes notification of the end of the process.

In this proposed architecture, process directions are usually sent from product agents. Focusing on processing or manufacturing products, there is no centralized control mechanism, and the proposed system becomes a flat architecture. Therefore, when the number of products or machine tools increases, a concentration in the load or messages to the central mechanism may occur in traditional hierarchical systems. But we can avoid such concentration in the proposed architecture.

2.2 Software Architecture

We introduce two software modules other than two agents described above. So, the proposed system uses up to four software modules involving the following:

- Product Agents
- Machine Agents
- Management Agents
- User interface program

The relationships between the software modules are shown in Fig.1. We will explain each module in the following sections.

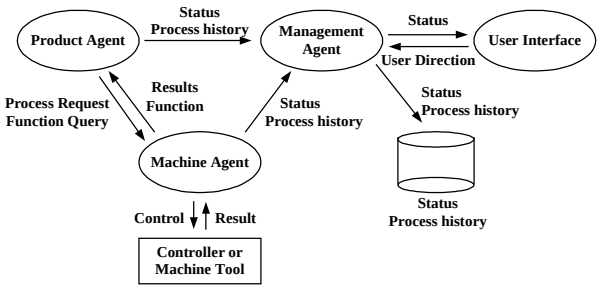

Fig. 1 Software Architecture

2.2.1 Product Agents

A product agent moves in computers or controllers following the movement of the target product. The product agent has a work order called a “recipe” and processes the product according to the recipe. Product agents have the responsibility to finish processing the target products.

A pair of { function name, command } is written on each line of the recipe. A product agent can complete manufacturing by processing the recipe line by line. The function name shows the name of the work process that each machine tool can provide. For example, “WASH” is the function name for the washing machine. The command shows the actual processing directions to the machine tool. The product agents search which machine tool can process the next recipe by sending messages which include the function name in the recipe as the search key. Then, the product agents select a machine agent among the candidates and direct the process specified in the command. Actually, the machine agents receive and reply to requests for selection or processing instead of the machine tools themselves. The selection process for the target machine tool or machine agent will be explained in detail in 2.3.

Product agents can monopolize one or more machine agents for exclusive control of using the machine tools. Exclusive control is performed by obtaining the access right to the machine agent. The product agent can have one or more access rights simultaneously. Only after a product agent gets the access rights for the target machine agents, the product agent send a request to the machine agent to carry out the process.

2.2.2 Machine Agent

Each machine agent corresponds to one machine tool. The machine agent has the responsibility to control a machine tool and grasps the status of the machine tool. The machine agent has the name of the machine in charge (e.g., CNC, WASH) as a function name. Several machine agents may have the same function name, for example, there are a couple of machine tools that are of the same type. The function names of the machine agents are specified in the recipe as explained before.

The machine agent can be monopolized simultaneously by one product agent at most and does the work sent from the product agent. The machine agent controls machine tools or controllers connected to the computer on which the machine agent is running and gets the status of the machine tools.

A machine agent can give an access right to at most one product agent for exclusive control to use the machine tool. Machine agents control the machine tool in charge by receiving directions from the product agent having the access right to the machine

agent only. Messages between the product agent and the machine agent are shown in Fig. 2.

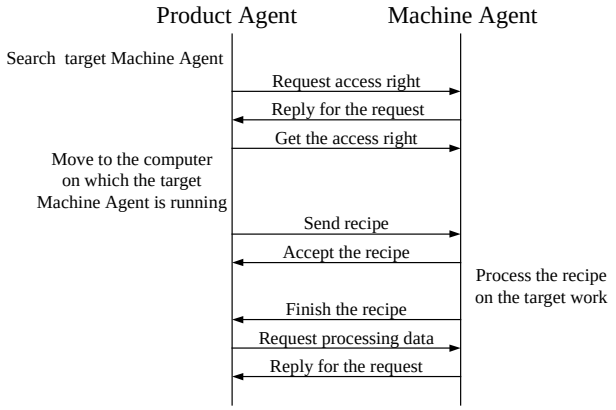

Fig. 2 Messages between Product Agent and Machine Agent

While the machine agent gives an access right to a product agent, new requests from other product agents are queued. After the product agent returns the access right, the machine agent gives a new access right by selecting one new product agent in the request queue. Product agents may send an access-right request to machine agents with priority. The priority is assigned depending on some criteria, for example, slack time to deadline of each product. In cases where one or more requests are queued on the machine agent, the machine agent selects one product agent with the highest priority request.

2.2.3 Management Agents

The management agent starts product agents and machine agents. Usually, machine agents are created at system startup, and product agents are created at the beginning of the actual processing of the target product. Product agents are created by system operators in the order of the production sequence.

Management agents receive production, quality, status data, or process histories sent from product and machine agents and save the data in persistent or nonvolatile storage. Management agents may also monitor the activities of product and machine agents.

2.2.4 User Interface Program

According to instructions made by an operator, the user interface program sends the operator's instructions to the product agents and machine agents. The user interface program can also display the states of each agent and each machine tool.

2.3 Selection of Target Machine Tool

We describe the procedure of selecting machine agents by product agents. This procedure is a kind of contract net. A product agent picks up the first line of the recipe. The product agent sends the function name and command in the recipe to the machine agents to search for which machine tools can process that recipe line. This message is sent by a broadcast message to all machine agents, or through group communications or by a multicast message to a specified machine group. Machine groups consist of machine agents having the same function name. Group communications is preferable in cases where there are a lot of machine tools, or in cases where scalability is required.

A machine agent that receives messages from the product agent decides whether the machine agent can accept the process request from the product agent. As a result, when the machine agent can do the requested process, the machine agent sends a reply message to the product agent that the machine agent can accept the request.

After the product agent receives the reply message, the product agent selects one target machine agent, that is, the target machine tool, among the machine agents which send the reply messages. Each product agent can have its own criteria to select the target machine agent. For example, a product agent can select the machine agent that sends the quickest reply message. The other product agent may consider the performance or availability of machine tools to select the target machine agent.

For machine availability, when a machine agent sends a reply message to a product agent, the machine agent can include a message whether or not the machine is being used at present. If the machine is not being used at present because it is being used for another process, the machine agent will apply a calculation using the presumed process time and find when the process can be started. This will enable the product agent to select the machine by which the process will start in as much as is possible. In addition, the product agent can select a machine agent enabling the recipe process to complete the most quickly.

After selecting the target machine agent, the product agent gets the access right first, and then sends an actual process request to the machine agent as shown in Fig.2.

3. Prototype System

To verify our concept and architecture, we developed a prototype system for the SDM Laboratory at CMU using the proposed architecture. In this section, we will first give a brief overview of the SDM Laboratory. Then, we will describe the hardware architecture and software implementation of the target system.

3.1 SDM Laboratory

The Shape Deposition Manufacturing (SDM) laboratory belongs to the Robotics Institute at Carnegie Mellon University. In SDM, the growing parts are built on pallets which are transferred from station-to-station using a robotic palletizing system. Each station has a pallet receiver mechanism. The parts-transfer robot places the pallet on the receiver which locates and clamps the pallet in place. The current SDM test-bed facility consists of four processing stations: CNC milling, deposition, shot-peening and cleaning. The deposition station also uses robotics to integrate multiple deposition processes.

3.2 Hardware Architecture

This system uses four main programmable controllers and two extra assistant controller stations. These main controllers are connected to industrial PCs (iPCs). The iPCs are connected to each other via ethernet. Each controller is also connected to the processing machines by a serial line. Each iPC has a Pentium 133MHz processor and 32MB memory.

There is also a supervisory or administrative PC used by an operator. The operator uses the supervisory PC to watch the system status and operates the machine tools. This supervisory PC has a Pentium Pro 200MHz processor and 64MB memory. Fig.3 shows the hardware architecture of the system.

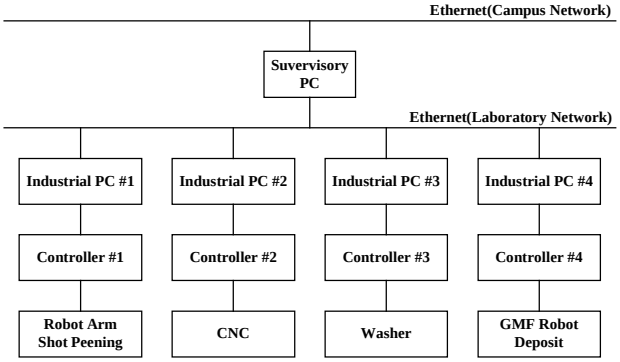

Fig. 3 Hardware Architecture

Each PC runs Windows NT 4.0. The local network of this system is separated from the CMU campus network in order to avoid outside influences while communicating between individual nodes in this system.

3.3 Implementation

We used Java as the implementation language and used native methods (JNI) to control the serial ports. We use aglets (alpha 5)[4,5] developed by IBM for movement of the agents and for communication between them. We also used HORB[6,7] for communication between the user interface and the management agent. JDK 1.1 was used at first for the Java VM environment, but a Symantec JIT Compiler is now being used because of its performance. The number of source code lines, excluding comments, is about 20,000. The number of classes excluding inner classes is 117. We used NTP on each PC in order to synchronize their clocks, and the supervisory PC worked as an NTP server.

Taking the system size into consideration, we selected broadcast messages for the product agents, aiming to search for the target machine agent. We did not take scalability into consideration in our prototype system. This is because the target system is for the closed laboratory and the system is small enough thanks to broadcast messages.

3.4 Availability

To increase availability in this prototype system, we used “I’m alive” messages. Both the product and machine agents sent their statuses to the management agent as “I’m alive” messages or heartbeat messages. The management agent watched the locations and statuses of both the product and machine agents, and also preserved the agents' logs on disks. In cases where the product or machine agent died, the management agent notified an operator, and started the product agent again, if necessary, after approval by the operator. But machine agents could not always recover from faults because we could not get the necessary information on the hardware status, or we could not redo the same process in the case of a machinery process.

3.5 Basic Performance

We show some performance results of our proposed system by implementing the prototype system. One evaluation concerns the performance of the movement of agents, and the other evaluation is related to some system performance.

These measurements were performed not in the actual prototype system, but in our experimental environment, because the actual system has already been in practical use. It does not influence the results to use our experimental environment rather than an actual system because we have to measure not the actual system performance, including the mechanical process time, but the computer system performance.

We used two PCs for measurements, one had a Pentium Pro(200MHz) processor and 64MB memory, the other PC had a Pentium(166MHz) processor and 48MB memory, and each machine was connected via ethernet. The operating system we used in these measurements was Windows NT 4.0. The network was isolated to avoid impact or interruptions from external networks. We used two Java execution environments for these measurements. One is Sun Microsystems JDK 1.1.5; the other is Symantec JIT Ver 3.0 (Visual Cafe 2.1).

3.5.1 Movements of agents

This section shows the amount of time for agents to move from one machine to another. The results are shown in Table 1.

Table. 1 Execution result of agent movements

Java VM	JDK1.1.5	JIT Ver.3.0
Round trip time	0.384(sec)	0.245(sec)

Each result in the table shows the round-trip time (in seconds) between two computers without class loading. The smaller number shows that its movement is faster. The number is an average of 500 operations. Each mobile agent is several hundred bytes in size.

This result indicates that the time it takes for an agent to move is faster than 1ms, and this time doesn't cause problems in the manufacturing process, because material handling and transfer time is usually much longer than the elapsed time for agent movement. We could not find any differences caused by class loading.

3.5.2 Execution of a recipe

Concerning the performance of recipe execution, we measured the amount of time for some lines of the recipe to execute. This means the time spent in the computers to process target work with the given recipe. The recipe for this measurement is:

1) PLACE WASH (Put the pallet on the washer)
2) RUN_FILE WASH WASH (Wash the pallet in the washer)
3) PICK WASH (Pick up the pallet from the washer)

We measured the execution time it took to process this recipe. We used machine 1 shown in Table 2.

Table. 2 Execution results of recipe

Java VM enviornent	JDK1.1.5	JIT Ver.3.0
Execution Time	27.78(sec)	16.35(sec)

4. Discussion

4.1 Flexibility of the Proposed System

Flexibility will be one of the most important issues to overcome in future manufacturing systems. Our proposed system provides the flexibility essential to future systems. We discuss the following requirements for the concerned flexibility.

- Changing system configuration
- Producing variable-type products
- Changing production order of products

We first describe the flexibility in changing the system configuration. In this system, product agents dynamically select a machine to process the next line of the recipe by using the function name in the line. In the prototype system, a product agent searches which machine agent can process a specified recipe by sending a broadcast message, while machine agents reply with their recipe processing availability to the product agent.

The product agent decides which machine to use according to the results of the search. Therefore, even in cases where some new machines are added, or some machines are taken out of service for maintenance or other reasons, product agents can select a suitable machine adapting to changes in the system configuration or the machines' operating status. Product agents need not know which machines are running; they can decide which machine to use by relying on only a reply message from the machine agents. Because the product agents and the machine agents work autonomously, they can change their behavior adapting the system configurations by themselves. The whole system can continue its activity coping with the configuration changes in the system in spite of no central control mechanism. In other words, we can add some new machines like Plug-and-Play flavor. In a case where the system has a central control mechanism, changes in the configuration or current status must be sent to the central control mechanism.

Next, we discuss the flexibility from changes in the target product. Each product agent has recipes (procedures) on how to produce the target product. So, it is easy to produce mixed and different types of products concurrently by changing only a recipe. We can build a system enabling the production of different types of products by giving a different recipe to each product agent accordingly. Changing recipes also makes a mixture of products with special required specifications.

This might make it possible to produce variable-type products in the central control mechanism. However, if the system configuration changes, or the types of products increase, an adjustment is required in the whole system. For this, the central mechanism needs to send a lot of messages to indicate the procedure changes to each controller. Whereas in our proposed system, each agent can act autonomously, the system enables variable types and variable amounts of production and can reduce the number of messages in the whole system.

Then, we discuss the flexibility involving changes in the order of production. In this system, each target product is mapped on each product agent. This can help complete urgent work or urgently needed products (with imminent delivery dates) quickly by changing the production order of the work through negotiation with the product agent that is in charge of the urgent work with other agents. In our proposed system, each product agent has a priority corresponding to the urgency of its handling product, and sends its priority to a machine agent with a monopolizing request. A machine agent selects a product agent as the next one having the highest priorities. This means a product agent whose priority is higher can take the access rights over product agents having a lower priority.

We evaluate the developed prototype system from the point of the required items for flexibility discussed above. Concerning changing system configuration, we can achieve this requirement to select the machine agents dynamically by the product agents. We can also change target products easily by changing a recipe given to the product agent, therefore we can produce variable-type products. On the other hand, concerning changing production order of products, we achieved this requirement to some extent in the prototype system by controlling request's priority, but we have to consider from other points of view.

4.2 System Architecture

When we consider a future manufacturing system with enough flexibility, it is required that there is no central management and that each subsystem work autonomously to avoid concentrating messages at some central mechanism. In this system, each product agent and machine agent can work autonomously and the system can continue its activities even if the management agent does not run. The product agents and the machine agents are key components in this system.

Our approach to using mobile agents for product agents is useful to avoid concentrating messages at several computers. If product agents are not mobile, they are collected on a specific computer to work, and if any excess load makes that computer overloaded, scalability will lack in the system architecture. In addition, if the specific computer on which the product agents work stops, the whole system will be affected.

A management agent is the only additional part to process target products or work and is for increasing dependability. The way adapted in the prototype system to increase dependability, that is, introducing the management agent, does not have scalability. In the case of a much larger system, other approaches, for example, introducing another messaging mechanism, are needed. If we use the agents in industrial fields, it is an important issue to ensure the activities of the agent's data, and this system ensures such requirements to some extent. However, in the practical use of the system, computer information has close ties to the status of the working product. Therefore, it is very difficult to restart the agents automatically. Confirmation by the system operator must be required.

5. Related Work

An approach that individual configuration element is considered as an active entity is also appeared in the concept of Holonic Manufacturing System[8], and an agent-oriented architecture for Holonic Manufacturing System was proposed[9]. Also, an agent-based approach for manufacturing system is proposed as in YAMS[10]. However,[9] and [10] describe the application of a multi-agent system in a conventional hierarchical control system. What are more, those agent's applications are used for scheduling. We can find that other studies in which agents are applied to the manufacturing or production system are focusing on scheduling in most cases. Our work, on the other hand, is targeting practical control fields rather than applying agents to the manufacturing

system for scheduling as in conventional studies. Our work in this paper focuses on how to handle processing products and machine tools in the agent-based system. From this point of view, our work is a new approach to a manufacturing system.

We are also proposing an autonomous decentralized system using agents rather than applying multiagents in a conventional hierarchical system. We believe that using this approach will provide a future manufacturing or production system with indispensable flexibility. We have not only proposed system architecture, but also developed a prototype system and run the system in practical use. We consider this application point of view to be one of our advanced activities.

In addition to the above, we can find a product or work oriented approach as in [11]. In [11], how to configure the system is not proposed; only simulation results are achieved. Compared to this, our system uses a concrete product oriented approach and activates actual systems thereby solving how to process variable-type products as well as how to handle machine tools in the system.

6. Conclusion and Future Work

We described our basic concept of a machinery control system using mobile agents in this paper. We also explained not only our concept but also system or software architecture of the actual prototype system. As a result of the above discussion, we showed our proposed concept and architecture to be useful in building flexible manufacturing systems. However, there are several problems to solve in future work.

We have to give more intelligence to the agents to increase the productivity or efficiency of the system. It is much better that the product agents are able to select the best paths or machines to satisfy objectives, for example, maximum productivity or no missed deadlines for all products, negotiating with other agents or avoiding deadlocks over some shared resources. The machine agent sends a reply message as to whether the received recipe can be processed or not in our prototype system. If the machine agent can send additional information involving the presumed process time as well as information of availability on machines, the product agent could find a more efficient path. Some cooperation with an external scheduling system is required in order to increase productivity in the whole entire system. These features have not yet been implemented in the prototype system. They are one of our future subjects to be solved.

Avoiding deadlocks over shared resources is one of important problems to solve because resource allocations are dynamically performed in our proposed system. We have proposed deadlock detection mechanism using some tokens.

Another important problem to solve is concerning dependability of the system. In the prototype system, some watching mechanism is implemented by using heartbeat signals. The product agent have relation to a physical product or work, and the machine agents also have relation to physical machine, so it would be one of the most important problems to ensure the agents activities. In the prototype system, we implemented the mechanism to increase availability of the system in application level, however, if agent platform could provide some mechanism, for example, check pointing and roll back function , we would use that mechanism.

Moreover, as we verify our concept on a small experimental system, we have to apply the proposed architecture to larger systems to determine whether the architecture is practical for actual systems.

7. ACKNOWLEDGMENTS

We would like to thank Prof. Lee Weiss and his laboratory's members for their support in building the prototype system in CMU. We are also grateful to Mr. Akira Nagashima, Vice President, and Mr. Tetuso Hoshi, General Manager, for giving us the opportunity of this work.

8. REFERENCES

[1] K. Mori, "Application in Rapidly Changing Environment," IEEE Computer, vol.31, no.4, pp.42-44, 1998.

[2] H. Wada, Y. Sakuraba, M. Negishi, T. Yamakawa, K. Kubo, and Y. Kashiyama, "A Machinery Control System Using Mobile Agents," Proc. 4th International Symposium on Autonomous Decentralized Systems, pp.124-131, March 1999.

[3] Shape Deposition Manufacturing Laboratory Home Page, http://www.cs.cmu.edu/~sdm.

[4] Aglets Software Development Kit Home Page, http://www.trl.ibm.co.jp/aglets/index.html.

[5] D. Lange and M.Oshima, "Programming and Deploying Java Mobil Agents with Aglets," Addison-Wesley, 1998.

[6] HORB Home Page, http://ring.etl.go.jp/openlab/horb.

[7] S. Hirano, "HORB: Distributed Execution of Java Programs," Worldwide Computing and Its Applications'97, Springer Lecture Notes in Computer Science 1274, pp29-42, 1997.

[8] D. Kriz, "Holonic Manufacturing Systems: Case Study of an IMS Consortium," http://hms.ifw.uni-hannover.de/, 1995.

[9] S. Bussmann, "An Agent-Oriented Architecture for Holonic Manufacturing Control," Proc. First International Workshop on IMS, pp.1-12, 1998.

[10] H.V.D. Parunak, B. Irish, J. Kindrich, and P. Lozo, "Fractal Actors for Distributed Manufacturing Control," Proc. 2nd Conference on AI Applications, pp.653-660, 1985.

[11] N. Gayed, D. Jarvis, and J. Jarvis, "A Strategy for the Migration of Existing Manufacturing Systems to Holonic Systems," Proc. IEEE Systems, Man, and Cybernetics Conf., San Diego, Oct. 1998.